Rajarata University of Sri Lanka

Department of Computing

ICT 2206 – Internet Programming

Laboratory 02 – Functions and Modular Programming using C++
Time: 120 Minutes

Intended Learning Outcomes (ILOs):

By the end of this practical, students should be able to:

1. Create an application with different activities implemented as different modules.
2. Use the different modules randomly in the application.
3. Use recursion to simplify the code.
4. Explain modular programming and reusable code.

Pre-Practical Preparation:

- **Reading/Resources**:
  - Students should refer to the lecture notes on computer programming and object-oriented programming.
  - Refer to https://www.w3schools.com/cpp/ for more information.

- **Preliminary Questions**:
  - What are functions in C++?
  - What are the two different types of functions in C++?
  - What is the use of the "`return`" statement in a C++ function?

Practical Objective:

To familiarise the students with the concept of modular programming and code reuse

Practical Activity:

Step-by-Step Guide/Instructions:

Step 1: Implement a function in the program to perform a simple task

- **Objective**: Learn how to implement functions in a program.

Instructions:

1. Open any text editor.
2. Type the following C++ code:

```
#include <iostream>
using namespace std;

int addNumbers(int a, int b)
{
     return a+b;
}

int main()
{
     int numOne, numTwo;
     cout << "Please enter the first number: ";
     cin >> numOne;
     cout << "Please enter the second number: ";
     cin >> numTwo;

     cout << "The sum is: " << addNumbers(numOne, numTwo) << endl;

     return 0;
}
```

3. Save the file with the name `addNum.cpp`.
4. Open the command prompt and change to the directory where you saved the file.
5. Compile the file using g++ by issuing the command `g++ addNum.cpp -o addNum.out`.
6. Run the code using the command `addNum.out`. and enter two numbers as the input.

Observation:

What do you see as the output?

Step 2: Implement a function that does not return a value

- **Objective**: Understand the use of a void function.

Instructions:

1. Type in the following code as a new file:

```
#include <iostream>
#include <string>
using namespace std;

void sayHello(string name)
{
     cout << "Hello " << name << ", Welcome!" << endl;
}
```

```
int main()
{
	string nameIn;
	cout << "Please enter your name: ";
	getline(cin, nameIn);
	sayHello(nameIn);

	return 0;
}
```

2. Save the file with the name `sayHello.cpp`.
3. Compile the file using g++ by issuing the command `g++ sayHello.cpp -o sayHello.out`.
4. Run the code using the command `sayHello.out` and enter your full name as the input.

Observation:

1. What do you see in the output?

Step 3: Implement a function that uses another function

- Objective: Learn how to invoke a function within another function.

Instructions:

1. Type in the following code as a new file:

```
#include <iostream>
using namespace std;

int addNumbers(int a, int b)
{
	return a+b;
}

float average(int x, int y)
{
	return (float)addNumbers(x,y)/2;
}

int main()
{
	int numOne, numTwo;
	cout << "Please enter the first number: ";
	cin >> numOne;
	cout << "Please enter the second number: ";
	cin >> numTwo;
	cout << "The sum is: " << addNumbers(numOne, numTwo) << endl;
	cout << "The average is: " << average(numOne, numTwo) << endl;

	return 0;
}
```

2. Save the file as `sumAve.cpp`.
3. Compile and run the program. Enter 23 and 22 as inputs.

Observation:

1. Do you see that the average value has a decimal?

Step 4: Use recursion in programming

- **Objective**: Learn how to call a function within itself.

Instructions:

1. Type in the following code as a new file:

```
#include <iostream>
using namespace std;

void findPrimeFactors(int n, int divisor = 2) {
    if (n == 1) {
        return;
    }

    if (n % divisor == 0) {
        cout << divisor << " ";
        findPrimeFactors(n / divisor, divisor);
    } else {
        findPrimeFactors(n, divisor + 1);
    }
}

int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;

    if (number <= 1) {
        cout << "Number must be greater than 1 to have factors." << endl;
    } else {
        cout << "Prime factors of " << number << ": ";
        findPrimeFactors(number);
        cout << endl;
    }

    return 0;
}
```

2. Save the file as `primeFactors.cpp`.
3. Compile and run the program. Enter a large number (<65,000) as the input.

Challenge:

Write a program to calculate the given term of the Fibonacci series. The number of the term must be input.

Write a *program to solve the Towers of Ha-Noi problem.*

Post-Practical Reflection:

- **Reflection Questions**:
    - What are the different types of functions defined in C++?
    - What are their uses?
- **Peer Discussion**: discuss with your friends and identify different scenarios where code reusing is useful.

Submission and Deadlines:

- **What to Submit**: a handwritten document containing the solution to the Towers of Ha-Noi problem.
